YB-60 Emulator Part 2

Landon
Lamoreaux
October 23rd, 2023

The program is run exactly as specified in the program document.

1. Starting the program with an object file as an argument will load that into the emulator.
2. Typing in the address will print the data at that location.
3. Typing in 2 addresses with a "." between them will print all the data between those 2 locations.
4. Typing in an address with a ":" then bytes with spaces between them will replace anything after that address with the data entered in.
5. Typing in an address followed by an "r" will run all the code starting at that address.
6. Typing in an address followed by a "t" will disassemble the code and print out the instructions that they once were.
7. Typing in "info" will print out the contents of every register.
8. You can exit the program by entering "exit" ctrl-C or ctrl-D.

The program imports and uses the following packages:

1. numpy
2. sys
3. re
4. BitArray from bitstring

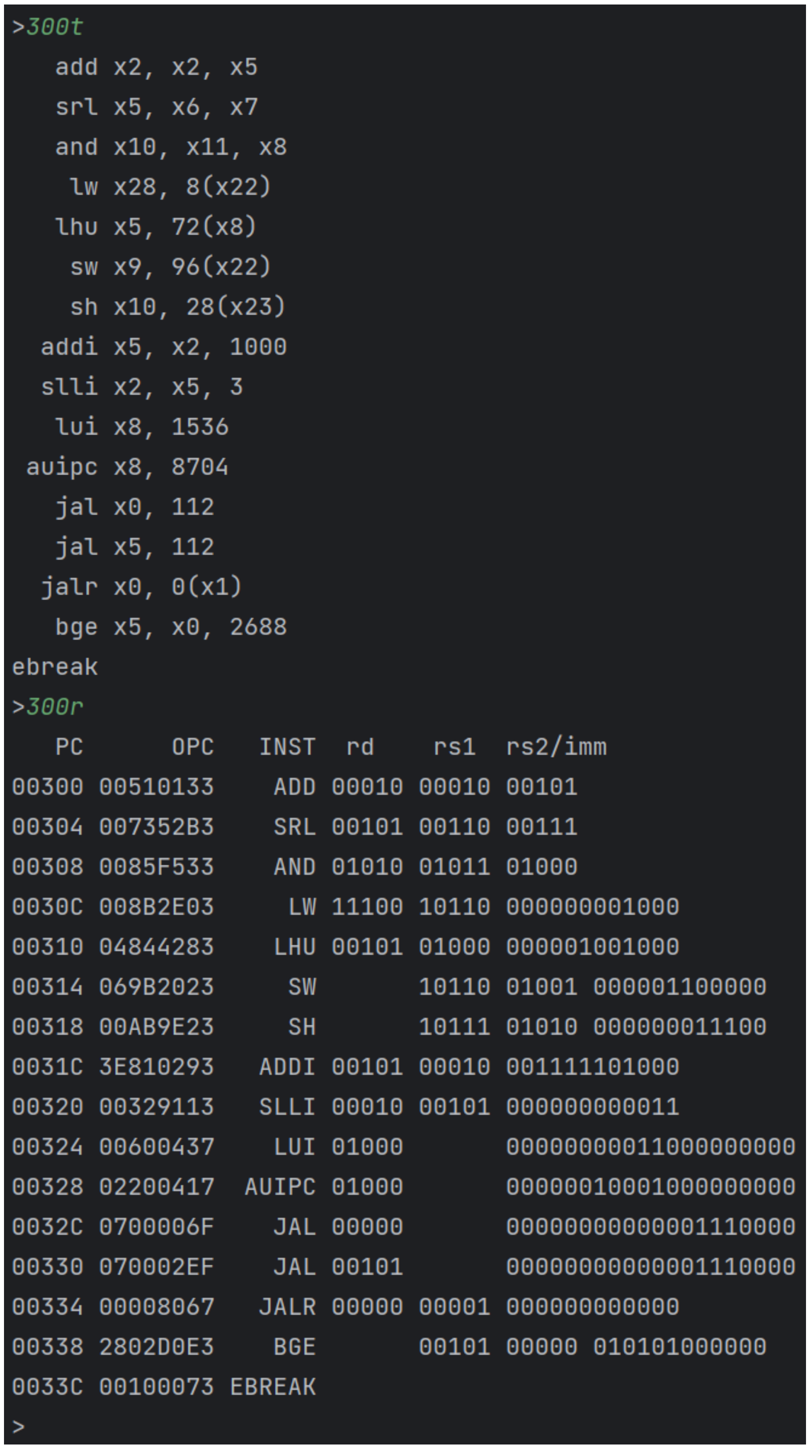
This emulator was developed in python 3.10.

If a user gives an address or data that is not in hexadecimal, then the program will output an error message and does not preform the function specified. It then prompts the user for new input.

Testing:

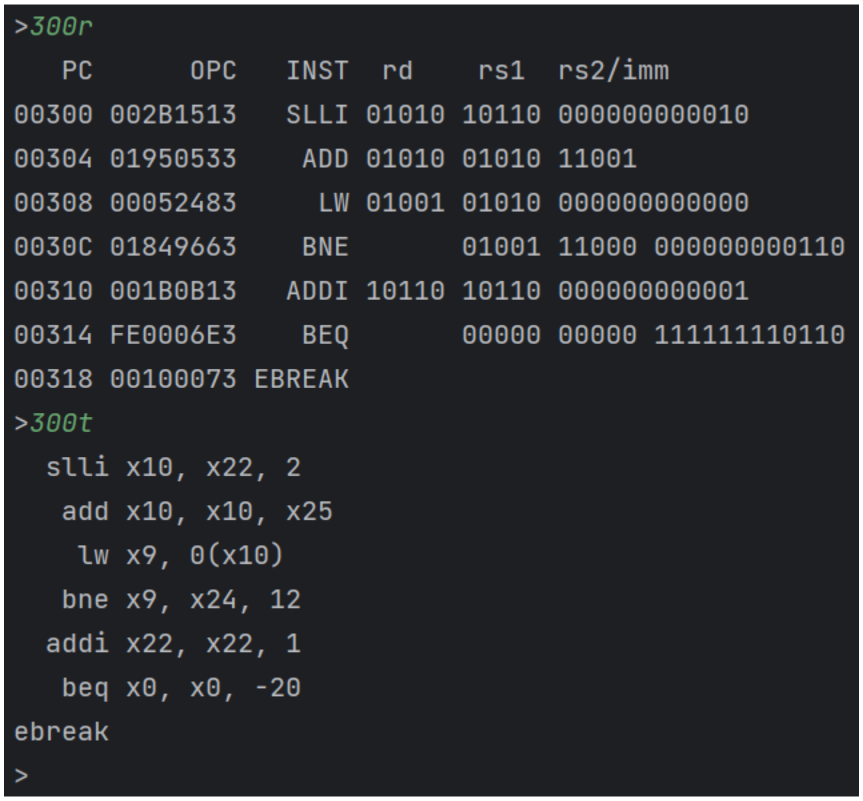
I tested the program by starting it with some of the provided object files, then running functions and comparing the output with the expected output that was provided for those files.

In the following image I ran the program with the file code.obj and then ran the disassembly function then the run function:



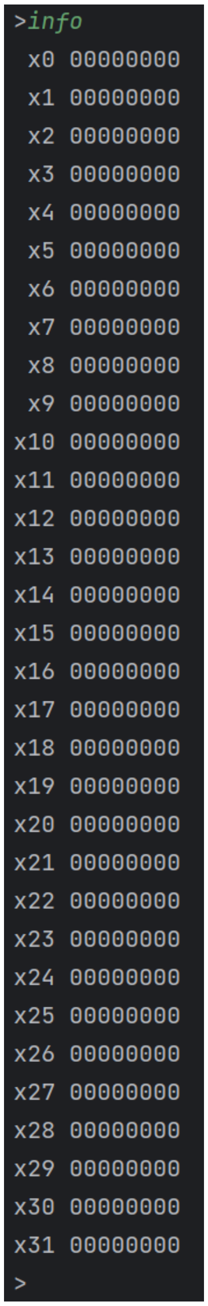
```
>300t
   add x2, x2, x5
   srl x5, x6, x7
   and x10, x11, x8
    lw x28, 8(x22)
   lhu x5, 72(x8)
    sw x9, 96(x22)
    sh x10, 28(x23)
  addi x5, x2, 1000
  slli x2, x5, 3
   lui x8, 1536
 auipc x8, 8704
   jal x0, 112
   jal x5, 112
  jalr x0, 0(x1)
   bge x5, x0, 2688
ebreak
>300r
   PC      OPC   INST  rd    rs1  rs2/imm
00300 00510133    ADD 00010 00010 00101
00304 007352B3    SRL 00101 00110 00111
00308 0085F533    AND 01010 01011 01000
0030C 008B2E03     LW 11100 10110 000000001000
00310 04844283    LHU 00101 01000 000001001000
00314 069B2023     SW       10110 01001 000001100000
00318 00AB9E23     SH       10111 01010 000000011100
0031C 3E810293   ADDI 00101 00010 001111101000
00320 00329113   SLLI 00010 00101 000000000011
00324 00600437    LUI 01000       00000000011000000000
00328 02200417  AUIPC 01000       00000010001000000000
0032C 0700006F    JAL 00000       00000000000001110000
00330 070002EF    JAL 00101       00000000000001110000
00334 00008067   JALR 00000 00001 000000000000
00338 2802D0E3    BGE       00101 00000 010101000000
0033C 00100073 EBREAK
>
```

In the next test I checked the output for the file ex2_7.obj:



```
>300r
   PC       OPC    INST  rd     rs1   rs2/imm
00300 002B1513   SLLI 01010 10110 000000000010
00304 01950533    ADD 01010 01010 11001
00308 00052483     LW 01001 01010 000000000000
0030C 01849663    BNE       01001 11000 000000000110
00310 001B0B13   ADDI 10110 10110 000000000001
00314 FE0006E3    BEQ       00000 00000 111111110110
00318 00100073 EBREAK
>300t
  slli x10, x22, 2
   add x10, x10, x25
    lw x9, 0(x10)
   bne x9, x24, 12
  addi x22, x22, 1
   beq x0, x0, -20
ebreak
>
```

Then I checked the output of running the “info” function:



```
>info
 x0 00000000
 x1 00000000
 x2 00000000
 x3 00000000
 x4 00000000
 x5 00000000
 x6 00000000
 x7 00000000
 x8 00000000
 x9 00000000
x10 00000000
x11 00000000
x12 00000000
x13 00000000
x14 00000000
x15 00000000
x16 00000000
x17 00000000
x18 00000000
x19 00000000
x20 00000000
x21 00000000
x22 00000000
x23 00000000
x24 00000000
x25 00000000
x26 00000000
x27 00000000
x28 00000000
x29 00000000
x30 00000000
x31 00000000
>
```